Amendments to the Claims

1. (Previously Presented) In a scheduler having at least one target processor, a method for ordering instructions in a code file having a plurality of instructions, the method comprising:

(a) determining dependencies between instructions in said plurality of instructions;

(b) creating a directed acyclic graph showing said dependencies in said plurality of instructions, where said directed acyclic graph's nodes each correspond to an instruction from said plurality of instructions;

(c) identifying one or more queues, including a first queue;

(d) traversing said directed acyclic graph in a dependency-preserving manner;

(e) creating a ready set of nodes comprising nodes in said directed acyclic graph having corresponding instructions in a ready state;

(f) finishing if there are no nodes having corresponding instructions in a ready state;

(g) identifying a threshold level in said first queue, said threshold corresponding to a maximum desirable fullness of said first queue;

(h) if said first queue is less full than said threshold, choosing a node in said ready set that corresponds to an instruction that would increase the fullness of said queue, if any such node exists in said ready set;

(i) if said first queue is at least as full as said threshold, choosing a node in said ready set that corresponds to an instruction that would decrease the fullness of said queue, if any such node exists in said ready set;

(j) if no node is chosen in (h) or (i), heuristically choosing a node in said ready set;

(k) removing said chosen node from said directed acyclic graph;

(l) modifying said first queue in accordance with said chosen node and its corresponding instruction;

(m) modifying the order of instructions in said code file in accordance with said chosen node and its corresponding instruction; and

(n) continuing processing at (d).

2. (Previously Presented) The scheduler of claim 1, wherein:

said first queue is configured to facilitate scheduling of the instructions;

said node chosen in (h) corresponds to an instruction that will add an element to said first queue; and

said node chosen in (i) corresponds to an instruction that will remove at least one element from said first queue.

3. (Cancelled)

4. (Previously Presented) The scheduler of claim 1, wherein said (c) comprises identifying a load queue, a prefetch queue and a store queue, the method further comprising:

(aa) determining and correlating a maximum desirable number of elements for each of said load queue, said prefetch queue, and said store queue; and

(bb) determining a precedence ordering between said load queue, said prefetch queue, and said store queue.

5. (Currently Amended) The scheduler of claim 1, wherein:

said (c) comprises:

(c1) determining a number of queues to use in accordance with a target processor;

(c2) determining a maximum desirable fullness for each of said determined number of queues; and

(c3) determining a precedence ordering between each of said determined number of queues; and

each of said (h) and said (i) comprises:

(hi1) choosing one of said nodes in said ready set that will change the number of elements in one of said determined number of queues, in accordance with said precedence order and said maximum desirable fullness, if one of said nodes can be found;

and

(hi2[[1]]) choosing one of said nodes in said ready set that will not change the number of elements in one of said determined number of queues, in accordance with said precedence order and said maximum desirable fullness, if none of said nodes in said ready set can be found that will change the number of elements in one of said determined number of queues.

6. (Previously Presented) The scheduler of claim 1, where said ordering of said instructions in said code file uses a hardware scheduler in accordance with said chosen node and its corresponding instruction.

7. (Previously Presented) The scheduler of claim 1, wherein said (l) comprises:

(aa) adding an element corresponding to said chosen node to said first queue if said first queue is less full than said threshold; and

(bb) removing at least one element in accordance with said chosen node from said first queue if said first queue is at least as full as said threshold.

8. (Previously Presented) A program storage device readable by a machine, tangibly embodying a program of instructions executable by the machine to perform a method for ordering instructions in a code file having a plurality of instructions, the method comprising:

(a) determining dependencies between instructions in said plurality of instructions;

(b) creating a directed acyclic graph showing said dependencies in said plurality of instructions, where said directed acyclic graph's nodes each correspond to an instruction from said plurality of instructions;

(c) identifying one or more queues, including a first queue;

(d) traversing said directed acyclic graph in a dependency-preserving manner;

(e) creating a ready set of nodes comprising nodes in said directed acyclic graph having corresponding instructions in a ready state;

(f) finishing if there are no nodes having corresponding instructions in a ready state;

(g) identifying a threshold level in said first queue, said threshold corresponding to a maximum desirable fullness of said first queue;

(h) if said first queue is less full than said threshold, choosing a node in said ready set that corresponds to an instruction that would increase the fullness of said queue, if any such node exists in said ready set;

(i) if said first queue is at least as full as said threshold, choosing a node in said ready set that corresponds to an instruction that would decrease the fullness of said queue, if any such node exists in said ready set;

(j) if no node is chosen in (h) or (i), heuristically choosing a node in said ready set;

(k) removing said chosen node from said directed acyclic graph;

(l) modifying said first queue in accordance with said chosen node and its corresponding instruction;

(m) modifying the order of instructions in said code file in accordance with said chosen node and its corresponding instruction; and

(n) continuing processing at (d).

9. (Previously Presented) The program storage device of claim 8, wherein:

said first queue is configured to facilitate scheduling of the instructions;

said node chosen in (h) corresponds to an instruction that will add an element to said first queue; and

said node chosen in (i) corresponds to an instruction that will remove at least one element from said first queue.

10. (Cancelled)

11. (Previously Presented) The program storage device of claim 8, wherein said (c) comprises identifying a load queue, a prefetch queue and a store queue,

the method further comprising:

(aa) determining and correlating a maximum desirable number of elements for each of said load queue, said prefetch queue, and said store queue; and

(bb) determining a precedence ordering between said load queue, said prefetch queue, and said store queue.

12. (Currently Amended) The program storage device of claim 8, wherein:

said (c) comprises:

(c1) determining a number of queues to use in accordance with a target processor;

(c2) determining a maximum desirable fullness for each of said determined number of queues; and

(c3) determining a precedence ordering between each of said determined number of queues; and

each of said (h) and said (i) comprises:

(hi1) choosing one of said nodes in said ready set that will change the number of elements in one of said determined number of queues, in accordance with said precedence order and said maximum desirable fullness, if one of said nodes can be found; and

(hi2[[1]]) choosing one of said nodes in said ready set that will not change the number of elements in one of said determined number of queues, in accordance with said precedence order and said maximum desirable fullness, if none of said nodes in said ready set can be found that will change the number of elements in one of said determined number of queues.

13. (Previously Presented) The program storage device of claim 8, where said ordering of said instructions in said code file uses a hardware scheduler in accordance with said chosen node and its corresponding instruction.

14. (Previously Presented) The program storage device of claim 8,

wherein said (l) comprises:

(aa) adding an element corresponding to said chosen node to said first queue if said first queue is less full than said threshold; and

(bb) removing at least one element in accordance with said chosen node from said first queue if said first queue is at least as full as said threshold.

15. (Currently Amended) A queue modeling instruction scheduler apparatus for use in compiling a program, the apparatus executable in a device having a processor operatively coupled to a memory, the apparatus comprising:

(a) a directed acyclic graph creation module configured to:

(a1) determine dependencies between instructions in a program to be compiled; and

(a2) create a directed acyclic graph showing said dependencies in said program, wherein said directed acyclic graph's nodes correspond to instructions in said program;

(b) a directed acyclic graph traversal and ready set identification module configured to:

(b1) traverse said directed acyclic graph in a dependency-preserving manner; and

(b2) create a ready set of nodes;

(c) a ready set evaluation module configured to:

(c1) identify which nodes in said ready set correspond to which instructions in said program; and

(c2) evaluate said instructions for their effect on memory operations;

(d) a queue management module configured to:

(d1) manage at least one queue, wherein managing a queue comprises adding and removing elements from said at least one queue, and wherein said elements correspond to nodes from said directed acyclic graph; and

(e) a code scheduling module operably connected to said program, said directed acyclic graph traversal and ready set identification module, said ready set evaluation module, said queue management module and said directed acyclic graph,

wherein said code scheduling module is configured to:

(e1) add and remove nodes to and from said directed acyclic graph;

(e2) determine and correlate a maximum desirable number of elements identifier for said at least one queue;

(e3) choose one of said nodes in said ready set that will change the number of elements in said at least one queue in accordance with said correlated identifier;

(e4[[3]]) have elements of said at least one queue added and removed; and

(e5[[4]]) change the order of instructions in said program in accordance with said instructions, said nodes, and said at least one queue.

16. (Previously Presented) The apparatus of claim 15, wherein said at least one queue comprises a load queue, a prefetch queue and a store queue, and wherein said code scheduling module is further configured to:

(e5) determine and correlate a maximum desirable number of elements identifier for each of said load queue, said prefetch queue, and said store queue;

(e6) determine a precedence ordering between said load queue, said prefetch queue, and said store queue; and

(e7) choose one of said nodes in said ready set that will change the number of elements in one of said load queue, said prefetch queue, or said store queue in accordance with said precedence order and one of said correlated identifiers.

17. (Previously Presented) The apparatus of claim 15, wherein:

said queue management module is further configured to:

(d2) determine a number of queues to use in accordance with a target processor; and

(d3) manage said determined number of queues; and

said code scheduling module is further configured to:

(e5) determine a precedence ordering between each of said determined number of queues; and

(e6) choose one of said nodes in said ready set that will affect the elements in one of said determined number of queues in accordance with said precedence order.

18. (Previously Presented) The apparatus of claim 15, wherein said code scheduler module comprises a hardware scheduler.

19. (Previously Presented) A method of scheduling a set of instructions during compilation of a program comprising the instructions, the method comprising:

(a) constructing a directed acyclic graph depicting dependencies among instructions in the set of instructions, wherein each node in the graph corresponds to an instruction in the set of instructions;

(b) identifying a ready set of nodes representing instructions having no dependencies on other instructions;

(c) identifying a target level of fullness within a queue configured to facilitate reordering of the set of instructions;

(d) if the queue is less full than said target level:

(d1) selecting a node in said ready set that corresponds to an instruction that would generate a memory operation; and

(d2) if said ready set includes no such node, heuristically selecting a node in said ready set;

(e) if the queue is at least as full as said target level:

(e1) selecting a node in said ready set that corresponds to an instruction that requires completion of a previous memory operation; and

(e2) if said ready set includes no such node, heuristically selecting a node in said ready set;

(f) removing the selected node from the ready set and the graph; and

(g) scheduling the instruction corresponding to the selected node.

20. (Previously Presented) The method of claim 19, wherein:

said instruction that would generate a memory operation comprises one of: a load,

a prefetch and a store; and

said instruction that requires completion of a previous memory operation is an instruction dependent upon one or more of: a load, a prefetch and a store.